Chapter 11

Markov Chain Monte Carlo Methods

11.1 Introduction

In many applications of statistical modeling, the data analyst would like to use a more complex model for a data set, but is forced to resort to an over-simplified model in order to use available techniques. Markov chain Monte Carlo (MCMC) methods are simulation-based and enable the statistician or engineer to examine data using realistic statistical models.

We start off with the following example taken from Raftery and Akman [1986] and Roberts [2000] that looks at the possibility that a change-point has occurred in a Poisson process. Raftery and Akman [1986] show that there is evidence for a change-point by determining Bayes factors for the change-point model versus other competing models. These data are a time series that indicate the number of coal mining disasters per year from 1851 to 1962. A plot of the data is shown in Figure 11.8, and it does appear that there has been a reduction in the rate of disasters during that time period. Some questions we might want to answer using the data are:

- What is the most likely year in which the change occurred?
- Did the rate of disasters increase or decrease after the change-point?

Example 11.8, presented later on, answers these questions using Bayesian data analysis and Gibbs sampling.

The main application of the MCMC methods that we present in this chapter is to generate a sample from a distribution. This sample can then be used to estimate various characteristics of the distribution such as moments, quantiles, modes, the density, or other statistics of interest.

In Section 11.2, we provide some background information to help the reader understand the concepts underlying MCMC. Because much of the recent developments and applications of MCMC arise in the area of Bayesian inference, we provide a brief introduction to this topic. This is followed by a discussion of Monte Carlo integration, since one of the applications of

MCMC methods is to obtain estimates of integrals. In Section 11.3, we present several Metropolis-Hastings algorithms, including the random-walk Metropolis sampler and the independence sampler. A widely used special case of the general Metropolis-Hastings method called the Gibbs sampler is covered in Section 11.4. An important consideration with MCMC is whether or not the chain has converged to the desired distribution. So, some convergence diagnostic techniques are discussed in Section 11.5. Sections 11.6 and 11.7 contain references to MATLAB code and references for the theoretical underpinnings of MCMC methods.

11.2 Background

Bayesian Inference

Bayesians represent uncertainty about unknown parameter values by probability distributions and proceed as if parameters were random quantities [Gilks, et al., 1996a]. If we let D represent the data that are observed and θ represent the model parameters, then to perform any inference, we must know the joint probability distribution $P(D, \theta)$ over all random quantities. Note that we allow θ to be multi-dimensional. From Chapter 2, we know that the joint distribution can be written as

$$P(D, \theta) = P(\theta)P(D|\theta),$$

where $P(\theta)$ is called the ***prior*** and $P(D|\theta)$ is called the ***likelihood***. Once we observe the data D, we can use Bayes' Theorem to get the ***posterior distribution*** as follows

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{\int P(\theta)P(D|\theta)d\theta}. \tag{11.1}$$

Equation 11.1 is the distribution of θ conditional on the observed data D. Since the denominator of Equation 11.1 is not a function of θ (since we are integrating over θ), we can write the posterior as being proportional to the prior times the likelihood,

$$P(\theta|D) \propto P(\theta)P(D|\theta) = P(\theta)L(\theta;D).$$

We can see from Equation 11.1 that the posterior is a conditional distribution for the model parameters given the observed data. Understanding and

using the posterior distribution is at the heart of Bayesian inference, where one is interested in making inferences using various features of the posterior distribution (e.g., moments, quantiles, etc.). These quantities can be written as posterior expectations of functions of the model parameters as follows

$$E[f(\theta)|D] = \frac{\int f(\theta)P(\theta)P(D|\theta)d\theta}{\int P(\theta)P(D|\theta)d\theta}. \tag{11.2}$$

Note that the denominator in Equations 11.1 and 11.2 is a constant of proportionality to make the posterior integrate to one. If the posterior is nonstandard, then this can be very difficult, if not impossible, to obtain. This is especially true when the problem is high dimensional, because there are a lot of parameters to integrate over. Analytically performing the integration in these expressions has been a source of difficulty in applications of Bayesian inference, and often simpler models would have to be used to make the analysis feasible. Monte Carlo integration using MCMC is one answer to this problem.

Because the same problem also arises in frequentist applications, we will change the notation to make it more general. We let $\boldsymbol{X}$ represent a vector of d random variables, with distribution denoted by $\pi(\mathbf{x})$. To a frequentist, $\boldsymbol{X}$ would contain data, and $\pi(\mathbf{x})$ is called a likelihood. For a Bayesian, $\boldsymbol{X}$ would be comprised of model parameters, and $\pi(\mathbf{x})$ would be called a posterior distribution. For both, the goal is to obtain the expectation

$$E[f(\boldsymbol{X})] = \frac{\int f(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}}{\int \pi(\mathbf{x})d\mathbf{x}}. \tag{11.3}$$

As we will see, with MCMC methods we only have to know the distribution of $\boldsymbol{X}$ up to the constant of normalization. This means that the denominator in Equation 11.3 can be unknown. It should be noted that in what follows we assume that $\boldsymbol{X}$ can take on values in a d-dimensional Euclidean space. The methods can be applied to discrete random variables with appropriate changes.

Monte Carlo Integration

As stated before, most methods in statistical inference that use simulation can be reduced to the problem of finding integrals. This is a fundamental part of the MCMC methodology, so we provide a short explanation of classical Monte Carlo integration. References that provide more detailed information on this subject are given in the last section of the chapter.

Monte Carlo integration estimates the integral $E[f(\boldsymbol{X})]$ of Equation 11.3 by obtaining samples $\boldsymbol{X}_t$, $t = 1, \ldots, n$ from the distribution $\pi(\mathbf{x})$ and calculating

$$E[f(\boldsymbol{X})] \approx \frac{1}{n}\sum_{t=1}^{n} f(\boldsymbol{X}_t)\,. \tag{11.4}$$

The notation t is used here because there is an ordering or sequence to the random variables in MCMC methods. We know that when the $\boldsymbol{X}_t$ are independent, then the approximation can be made as accurate as needed by increasing n. We will see in the following sections that with MCMC methods, the samples are not independent in most cases. That does not limit their use in finding integrals using approximations such as Equation 11.4. However, care must be taken when determining the variance of the estimate in Equation 11.4 because of dependence [Gentle, 1998; Robert and Casella, 1999]. We illustrate the method of Monte Carlo integration in the next example.

Example 11.1

For a distribution that is exponential with $\lambda = 1$, we find $E[\sqrt{X}]$ using Equation 11.4. We generate random variables from the required distribution, take the square root of each one and then find the average of these values. This is implemented below in MATLAB.

```
% Generate 500 exponential random
% variables with lambda = 1.
% This is a Statistics Toolbox function.
x = exprnd(1,1,1000);
% Take square root of each one.
xroot = sqrt(x);
% Take the mean - Equation 11.4
exroothat = mean(xroot);
```

From this, we get an estimate of 0.889. We can use MATLAB to find the value using numerical integration.

```
% Now get it using numerical integration
strg = 'sqrt(x).*exp(-x)';
myfun = inline(strg);
% quadl is a MATLAB 6 function.
exroottru = quadl(myfun,0,50);
```

The value we get using numerical integration is 0.886, which closely matches what we got from the Monte Carlo method.

❑

The samples $\boldsymbol{X}_t$ do not have to be independent as long as they are generated using a process that obtains samples from the *'entire'* domain of $\pi(\mathbf{x})$ and in the correct proportions [Gilks, et al., 1996a]. This can be done by constructing a Markov chain that has $\pi(\mathbf{x})$ as its stationary distribution. We now give a brief description of Markov chains.

Markov Chains

A Markov chain is a sequence of random variables such that the next value or state of the sequence depends only on the previous one. Thus, we are generating a sequence of random variables, $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots$ such that the next state $\boldsymbol{X}_{t+1}$ with $t \geq 0$ is distributed according to $P(\boldsymbol{X}_{t+1} | \boldsymbol{X}_t)$, which is called the ***transition kernel***. A realization of this sequence is also called a Markov chain. We assume that the transition kernel does not depend on t, making the chain time-homogeneous.

One issue that must be addressed is how sensitive the chain is to the starting state $\boldsymbol{X}_0$. Given certain conditions [Robert and Casella, 1999], the chain will forget its initial state and will converge to a stationary distribution, which is denoted by ψ. As the sequence grows larger, the sample points $\boldsymbol{X}_t$ become dependent samples from ψ. The reader interested in knowing the conditions under which this happens and for associated proofs of convergence to the stationary distribution is urged to read the references given in Section 11.7.

Say the chain has been run for m iterations, and we can assume that the sample points $\boldsymbol{X}_t$, $t = m+1, \ldots, n$ are distributed according to the stationary distribution ψ. We can discard the first m iterations and use the remaining $n - m$ samples along with Equation 11.4 to get an estimate of the expectation as follows

$$E[f(\boldsymbol{X})] \approx \frac{1}{n-m} \sum_{t=m+1}^{n} f(\boldsymbol{X}_t) . \tag{11.5}$$

The number of samples m that are discarded is called the ***burn-in***. The size of the burn-in period is the subject of current research in MCMC methods. Diagnostic methods to help determine m and n are described in Section 11.5. Geyer [1992] suggests that the burn-in can be between 1% and 2% of n, where n is large enough to obtain adequate precision in the estimate given by Equation 11.5.

So now we must answer the question: how large should n be to get the required precision in the estimate? As stated previously, estimating the variance of the estimate given by Equation 11.5 is difficult because the samples are not independent. One way to determine n via simulation is to run several Markov chains in parallel, each with a different starting value. The estimates from Equation 11.5 are compared, and if the variation between them is too

great, then the length of the chains should be increased [Gilks, et al., 1996b]. Other methods are given in Roberts [1996], Raftery and Lewis [1996], and in the general references mentioned in Section 11.7.

Analyzing the Output

We now discuss how the output from the Markov chains can be used in statistical analysis. An analyst might be interested in calculating means, standard deviations, correlations and marginal distributions for components of $\boldsymbol{X}$. If we let $X_{t,j}$ represent the j-th component of $\boldsymbol{X}_t$ at the t-th step in the chain, then using Equation 11.5, we can obtain the marginal means and variances from

$$\overline{X}_{,j} = \frac{1}{n-m} \sum_{t=m+1}^{n} X_{t,j},$$

and

$$S^2_{,j} = \frac{1}{n-m-1} \sum_{t=m+1}^{n} (X_{t,j} - \overline{X}_{,j})^2.$$

These estimates are simply the componentwise sample mean and sample variance of the sample points $\boldsymbol{X}_t$, $t = m+1, \ldots, n$. Sample correlations are obtained similarly. Estimates of the marginal distributions can be obtained using the techniques of Chapter 8.

One last problem we must deal with to make Markov chains useful is the stationary distribution ψ. We need the ability to construct chains such that the stationary distribution of the chain is the one we are interested in: $\pi(\mathbf{x})$. In the MCMC literature, $\pi(\mathbf{x})$ is often referred to as the ***target distribution***. It turns out that this is not difficult and is the subject of the next two sections.

11.3 Metropolis-Hastings Algorithms

The Metropolis-Hastings method is a generalization of the Metropolis technique of Metropolis, et al. [1953], which had been used for many years in the physics community. The paper by Hastings [1970] further generalized the technique in the context of statistics. The Metropolis sampler, the independence sampler and the random-walk are all special cases of the Metropolis-

Hastings method. Thus, we cover the general method first, followed by the special cases.

These methods share several properties, but one of the more useful properties is that they can be used in applications where $\pi(\mathbf{x})$ is known up to the constant of proportionality. Another property that makes them useful in a lot of applications is that the analyst does not have to know the conditional distributions, which is the case with the Gibbs sampler. While it can be shown that the Gibbs sampler is a special case of the Metropolis-Hastings algorithm [Robert and Casella, 1999], we include it in the next section because of this difference.

Metropolis-Hastings Sampler

The Metropolis-Hastings sampler obtains the state of the chain at $t+1$ by sampling a ***candidate point*** $\boldsymbol{Y}$ from a ***proposal distribution*** $q(.|\boldsymbol{X}_t)$. Note that this depends only on the previous state $\boldsymbol{X}_t$ and can have any form, subject to regularity conditions [Roberts, 1996]. An example for $q(.|\boldsymbol{X}_t)$ is the multivariate normal with mean $\boldsymbol{X}_t$ and fixed covariance matrix. One thing to keep in mind when selecting $q(.|\boldsymbol{X}_t)$ is that the proposal distribution should be easy to sample from.

The required regularity conditions for $q(.|\boldsymbol{X}_t)$ are irreducibility and aperiodicity [Chib and Greenberg, 1995]. ***Irreducibility*** means that there is a positive probability that the Markov chain can reach any non-empty set from all starting points. ***Aperiodicity*** ensures that the chain will not oscillate between different sets of states. These conditions are usually satisfied if the proposal distribution has a positive density on the same support as the target distribution. They can also be satisfied when the target distribution has a restricted support. For example, one could use a uniform distribution around the current point in the chain.

The candidate point is accepted as the next state of the chain with probability given by

$$\alpha(\boldsymbol{X}_t, \boldsymbol{Y}) = \min\left\{1, \frac{\pi(\boldsymbol{Y})q(\boldsymbol{X}_t|\boldsymbol{Y})}{\pi(\boldsymbol{X}_t)q(\boldsymbol{Y}|\boldsymbol{X}_t)}\right\}. \tag{11.6}$$

If the point $\boldsymbol{Y}$ is not accepted, then the chain does not move and $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t$. The steps of the algorithm are outlined below. It is important to note that the distribution of interest $\pi(\mathbf{x})$ appears as a ratio, so the constant of proportionality cancels out. This is one of the appealing characteristics of the Metropolis-Hastings sampler, making it appropriate for a wide variety of applications.

PROCEDURE - METROPOLIS-HASTINGS SAMPLER

1. Initialize the chain to $\boldsymbol{X}_0$ and set $t = 0$.
2. Generate a candidate point $\boldsymbol{Y}$ from $q(.|\boldsymbol{X}_t)$.
3. Generate U from a uniform $(0, 1)$ distribution.
4. If $U \leq \alpha(\boldsymbol{X}_t, Y)$ (Equation 11.6) then set $\boldsymbol{X}_{t+1} = \boldsymbol{Y}$, else set $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t$.
5. Set $t = t + 1$ and repeat steps 2 through 5.

The Metropolis-Hastings procedure is implemented in Example 11.2, where we use it to generate random variables from a standard Cauchy distribution. As we will see, this implementation is one of the special cases of the Metropolis-Hastings sampler described later.

Example 11.2

We show how the Metropolis-Hastings sampler can be used to generate random variables from a standard Cauchy distribution given by

$$f(x) = \frac{1}{\pi(1 + x^2)}; \qquad -\infty < x < \infty.$$

From this, we see that

$$f(x) \propto \frac{1}{1 + x^2}.$$

We will use the normal as our proposal distribution, with a mean given by the previous value in the chain and a standard deviation given by σ. We start by setting up **inline** MATLAB functions to evaluate the densities for Equation 11.6.

```
% Set up an inline function to evaluate the Cauchy.
% Note that in both of the functions,
% the constants are canceled.
strg = '1./(1+x.^2)';
cauchy = inline(strg,'x');
% set up an inline function to evaluate the Normal pdf
strg = '1/sig*exp(-0.5*((x-mu)/sig).^2)';
norm = inline(strg,'x','mu','sig');
```

We now generate $n = 10000$ samples in the chain.

```
% Generate 10000 samples in the chain.
% Set up the constants.
n = 10000;
```

```
sig = 2;
x = zeros(1,n);
x(1) = randn(1);% generate the starting point
for i = 2:n
 % generate a candidate from the proposal distribution
 % which is the normal in this case. This will be a
 % normal with mean given by the previous value in the
 % chain and standard deviation of 'sig'
 y = x(i-1) + sig*randn(1);
 % generate a uniform for comparison
 u = rand(1);
 alpha = min([1, cauchy(y)*norm(x(i-1),y,sig)/...
         (cauchy(x(i-1))*norm(y,x(i-1),sig))]);
 if u <= alpha
   x(i) = y;
 else
   x(i) = x(i-1);
 end
end
```

We can plot a density histogram along with the curve corresponding to the true probability density function. We discard the first 500 points for the burn-in period. The plot is shown in Figure 11.1.
❑

Metropolis Sampler

The Metropolis sampler refers to the original method of Metropolis, et al. [1953], where only symmetric distributions are considered for the proposal distribution. Thus, we have that

$$q(Y|X) = q(X|Y).$$

for all X and Y. As before, a common example of a distribution like this is the normal distribution with mean X and fixed covariance. Because the proposal distribution is symmetric, those terms cancel out in the acceptance probability yielding

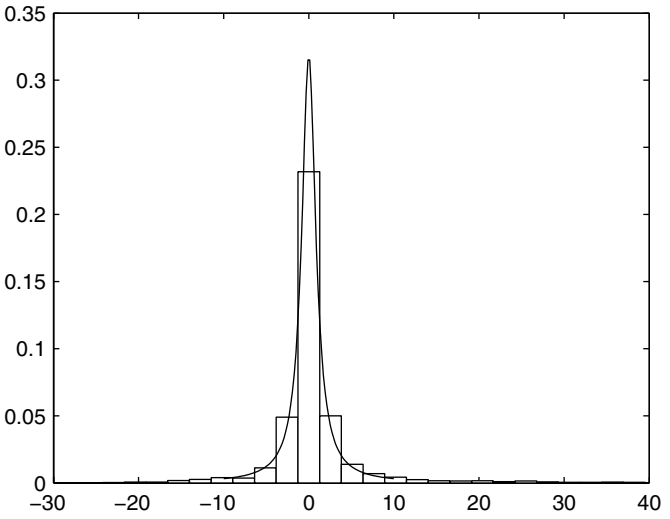

FIGURE 11.1
We generated 10,000 variates from the Cauchy distribution using the Metropolis-Hastings sampler. This shows a density histogram of the random variables after discarding the first 500 points. The curve corresponding to the true probability density function is superimposed over the histogram. We see that the random variables do follow the standard Cauchy distribution.

$$\alpha(\boldsymbol{X}_t, \boldsymbol{Y}) = \min\left\{1, \frac{\pi(\boldsymbol{Y})}{\pi(\boldsymbol{X}_t)}\right\}. \tag{11.7}$$

PROCEDURE - METROPOLIS SAMPLER

1. Initialize the chain to $\boldsymbol{X}_0$ and set $t = 0$.
2. Generate a candidate point $\boldsymbol{Y}$ from $q(.|\boldsymbol{X}_t)$.
3. Generate U from a uniform $(0, 1)$ distribution.
4. If $U \leq \alpha(\boldsymbol{X}_t, \boldsymbol{Y})$ (Equation 11.7) then set $\boldsymbol{X}_{t+1} = \boldsymbol{Y}$, else set $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t$.
5. Set $t = t + 1$ and repeat steps 2 through 5.

When the proposal distribution is such that $q(\boldsymbol{Y}|\boldsymbol{X}) = q(|\boldsymbol{X} - \boldsymbol{Y}|)$, then it is called the ***random-walk Metropolis***. This amounts to generating a candidate

point $\boldsymbol{Y} = \boldsymbol{X}_t + \boldsymbol{Z}$, where $\boldsymbol{Z}$ is an increment random variable from the distribution q.

We can gain some insight into how this algorithm works by looking at the conditions for accepting a candidate point as the next sample in the chain. In the symmetric case, the probability of moving is $\pi(\boldsymbol{Y})/\pi(\boldsymbol{X}_t)$. If $\pi(\boldsymbol{Y}) \geq \pi(\boldsymbol{X}_t)$, then the chain moves to $\boldsymbol{Y}$ because $\alpha(\boldsymbol{X}_t, \boldsymbol{Y})$ will be equal to 1. This means that a move that climbs up the curve given by the target distribution is always accepted. A move that is worse (i.e., one that goes downhill) is accepted with probability given by $\pi(\boldsymbol{Y})/\pi(\boldsymbol{X}_t)$. These concepts are illustrated in Figure 11.2. This is the basic algorithm proposed by Metropolis, et al. [1953], and it is the foundation for other optimization algorithms such as simulated annealing [Kirkpatrick, Gelatt, and Vechi, 1983; Aarts and Korst, 1989].

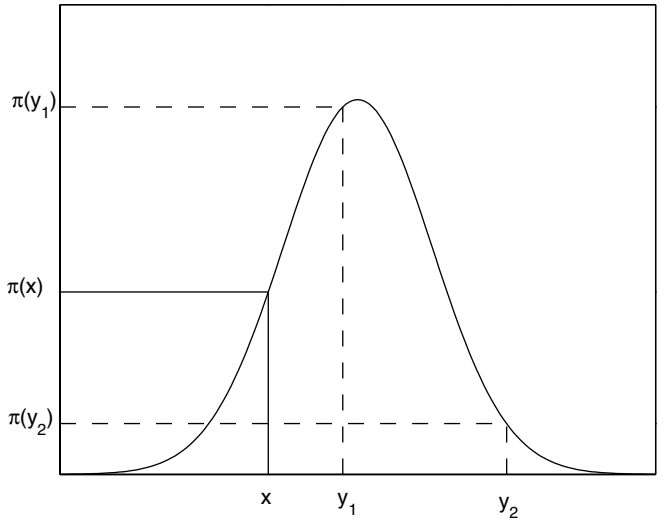

FIGURE 11.2
This shows what happens when a candidate point is selected and the proposal distribution is symmetric [Chib and Greenberg, 1995]. In this case, the probability of moving to another point is based on the ratio $\pi(y)/\pi(x)$. If $\pi(y) \geq \pi(x)$, then the chain moves to the candidate point y. If $\pi(y) < \pi(x)$, then the chain moves to y with probability $\pi(y)/\pi(x)$. So we see that a move from x to y_1 would be automatically accepted, but a move to y_2 would be accepted with probability $\pi(y_2)/\pi(x)$.

When implementing any of the Metropolis-Hastings algorithms, it is important to understand how the scale of the proposal distribution affects the efficiency of the algorithm. This is especially apparent with the random-walk version and is illustrated in the next example. If a proposal distribution takes small steps, then the acceptance probability given by Equation 11.7 will be

high, yielding a higher rate at which we accept candidate points. The problem here is that the chain will mix slowly, meaning that the chain will take longer to get to the stationary distribution. On the other hand, if the proposal distribution generates large steps, then the chain could move to the tails, resulting in low acceptance probabilities. Again, the chain fails to mix quickly.

Example 11.3

In this example, we show how to implement the random-walk version of the Metropolis-Hastings sampler [Gilks, et al., 1996a] and use it to generate variates from the standard normal distribution (the target distribution). Of course, we do not have to resort to MCMC methods to generate random variables from this target distribution, but it serves to illustrate the importance of picking the right scale for the proposal distribution. We use the normal as a proposal distribution to generate the candidates for the next value in the chain. The mean of the proposal distribution is given by the current value in the chain x_t. We generate three chains with different values for the standard deviation, given by: $\sigma = 0.5, 0.1, 10$. These provide chains that exhibit good mixing, poor mixing due to small step size and poor mixing due to a large step size, respectively. We show below how to generate the three sequences with $n = 500$ variates in each chain.

```
% Get the variances for the proposal distributions.
sig1 = 0.5;
sig2 = 0.1;
sig3 = 10;
% We will generate 500 iterations of the chain.
n = 500;
% Set up the vectors to store the samples.
X1 = zeros(1,n);
X2 = X1;
X3 = X1;
% Get the starting values for the chains.
X1(1) = -10;
X2(1) = 0;
X3(1) = 0;
```

Now that we have everything initialized, we can obtain the chains.

```
% Run the first chain.
for i = 2:n
 % Generate variate from proposal distribution.
 y = randn(1)*sig1 + X1(i-1);
 % Generate variate from uniform.
 u = rand(1);
 % Calculate alpha.
```

```
 alpha = normpdf(y,0,1)/normpdf(X1(i-1),0,1);
 if u <= alpha
   % Then set the chain to the y.
   X1(i) = y;
 else
   X1(i) = X1(i-1);
 end
end
% Run second chain.
for i = 2:n
 % Generate variate from proposal distribution.
 y = randn(1)*sig2 + X2(i-1);
 % Generate variate from uniform.
 u = rand(1);
 % Calculate alpha.
 alpha = normpdf(y,0,1)/normpdf(X2(i-1),0,1);
 if u <= alpha
   % Then set the chain to the y.
   X2(i) = y;
 else
   X2(i) = X2(i-1);
 end
end
% Run the third chain.
for i = 2:n
 % Generate variate from proposal distribution.
 y = randn(1)*sig3 + X3(i-1);
 % Generate variate from uniform.
 u = rand(1);
 % Calculate alpha.
 alpha = normpdf(y,0,1)/normpdf(X3(i-1),0,1);
 if u <= alpha
   % Then set the chain to the y.
   X3(i) = y;
 else
   X3(i) = X3(i-1);
 end
end
```

Plots of these sequences are illustrated in Figure 11.3, where we also show horizontal lines at ± 2. These lines are provided as a way to determine if most values in the chain are mixing well (taking on many different values) within two standard deviations of zero, since we are generating standard normal variates. Note that the first one converges quite rapidly and exhibits good mixing in spite of an extreme starting point. The second one with $\sigma = 0.1$ (small steps) is mixing very slowly and does not seem to have converged to

the target distribution in these 500 steps of the chain. The third sequence, where large steps are taken, also seems to be mixing slowly, and it is easy to see that the chain sometimes does not move. This is due to the large steps taken by the proposal distribution.
❑

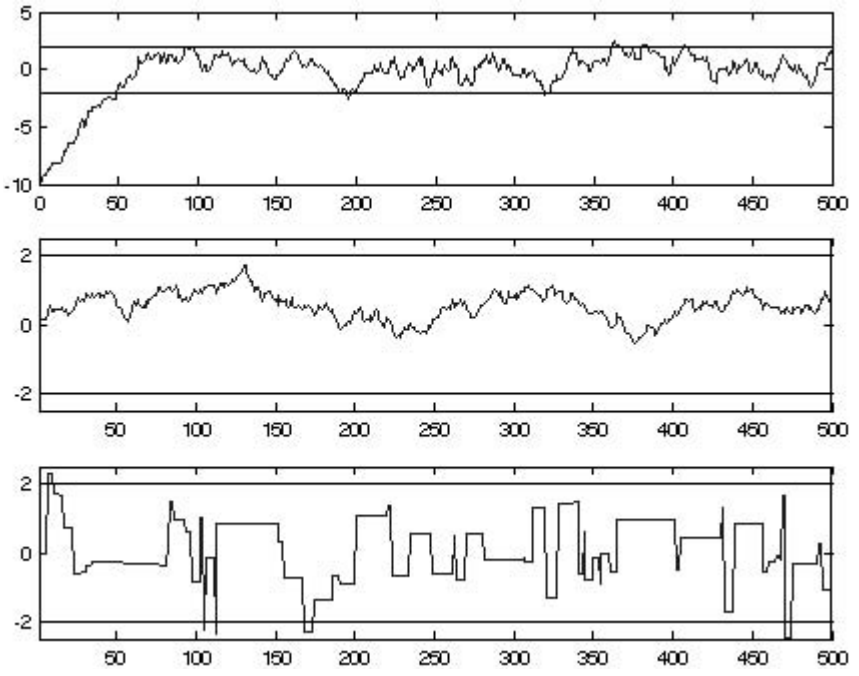

FIGURE 11.3
These are the three sequences from Example 11.3. The target distribution is the standard normal. For all three sequences, the proposal distribution is normal with the mean given by the previous element in the sequence. The standard deviations of the proposal distribution are: $\sigma = 0.5, 0.1, 10$. Note that the first sequence approaches the target distribution after the first 50 - 100 iterations. The other two sequences are slow to converge to the target distribution because of slow mixing due to the poor choice of σ.

Independence Sampler

The independence sampler was proposed by Tierney [1994]. This method uses a proposal distribution that does not depend on $\boldsymbol{X}$; i.e., it is generated independently of the previous value in the chain. The proposal distribution is of the form $q(Y|\boldsymbol{X}) = q(Y)$, so Equation 11.6 becomes

$$\alpha(\boldsymbol{X}_t, Y) = \min\left\{1, \frac{\pi(Y)q(\boldsymbol{X}_t)}{\pi(\boldsymbol{X}_t)q(Y)}\right\}. \tag{11.8}$$

This is sometimes written in the literature as

$$\alpha(\boldsymbol{X}_t, Y) = \min\left\{1, \frac{w(Y)}{w(\boldsymbol{X}_t)}\right\}$$

where $w(\boldsymbol{X}) = \pi(\boldsymbol{X})/q(\boldsymbol{X})$.

Caution should be used when implementing the independence sampler. In general, this method will not work well unless the proposal distribution q is very similar to the target distribution π. Gilks, et al. [1996a] show that it is best if q is heavier-tailed than π. Note also that the resulting sample is still not independent, even though we generate the candidate points independently of the previous value in the chain. This is because the acceptance probability for the next value $\boldsymbol{X}_{t+1}$ depends on the previous one. For more information on the independence sampler and the recommended usage, see Roberts [1996] or Robert and Casella [1999].

Autoregressive Generating Density

Another choice for a candidate generating density is proposed by Tierney [1994] and described by Chib and Greenberg [1995]. This is represented by an autoregressive process of order 1 and is obtained by generating candidates as follows

$$\boldsymbol{Y} = \boldsymbol{a} + \mathbf{B}(\boldsymbol{X}_t - \boldsymbol{a}) + \boldsymbol{Z}, \tag{11.9}$$

where $\boldsymbol{a}$ is a vector and $\mathbf{B}$ is a matrix, both of which are conformable in terms of size with $\boldsymbol{X}_t$. The vector $\boldsymbol{Z}$ has a density given by q. If $\mathbf{B} = -\mathbf{I}$, then the chains are produced by reflecting about the point $\boldsymbol{a}$, yielding negative correlation between successive values in the sequence. The autoregressive generating density is described in the next example.

Example 11.4

We show how to use the Metropolis-Hastings sampler with the autoregressive generating density to generate random variables from a target distribution given by a bivariate normal with the following parameters:

$$\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}.$$

Variates from this distribution can be easily generated using the techniques of Chapter 4, but it serves to illustrate the concepts. In the exercises, the reader is asked to generate a set of random variables using those techniques and compare them to the results obtained in this example. We generate a sequence of $n = 6000$ points and use a burn-in of 4000.

```
% Set up some constants and arrays to store things.
n = 6000;
xar = zeros(n,2); % to store samples
mu = [1;2];  % Parameters - target distribution.
covm = [1 0.9; 0.9 1];
```

We now set up a MATLAB **inline** function to evaluate the required probabilities.

```
% Set up the function to evaluate alpha
% for this problem. Note that the constant
% has been canceled.
strg = 'exp(-0.5*(x-mu)''*inv(covm)*(x-mu))';
norm = inline(strg,'x','mu','covm');
```

The following MATLAB code sets up a random starting point and obtains the elements of the chain.

```
% Generate starting point.
xar(1,:) = randn(1,2);
for i = 2:n
 % Get the next variate in the chain.
 % y is a column vector.
 y = mu - (xar(i-1,:)'-mu) + (-1+2*rand(2,1));
 u = rand(1);
 % Uses inline function 'norm' from above.
 alpha=min([1,norm(y,mu,covm)/...
         norm(xar(i-1,:)',mu,covm)]);
 if u <= alpha
   xar(i,:) = y';
 else
   xar(i,:) = xar(i-1,:);
 end
end
```

A scatterplot of the last 2000 variates is given in Figure 11.4, and it shows that they do follow the target distribution. To check this further, we can get the sample covariance matrix and the sample mean using these points. The result is

$$\hat{\boldsymbol{\mu}} = \begin{bmatrix} 1.04 \\ 2.03 \end{bmatrix} \qquad \hat{\boldsymbol{\Sigma}} = \begin{bmatrix} 1 & 0.899 \\ 0.899 & 1 \end{bmatrix},$$

from which we see that the sample does reflect the target distribution.
❑

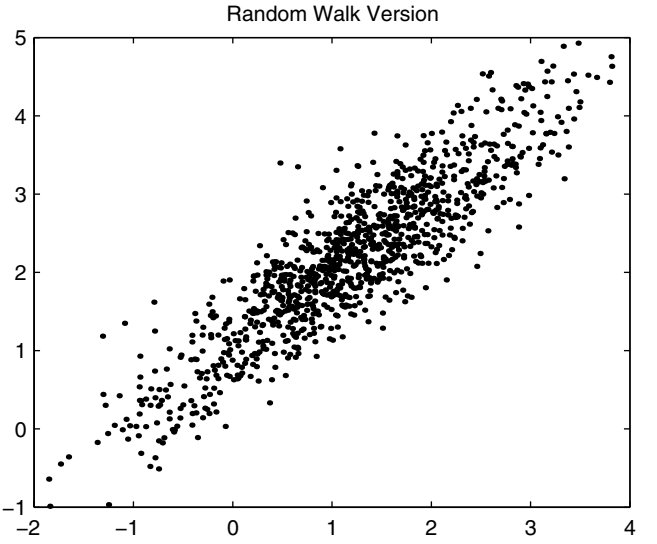

FIGURE 11.4
This is a scatterplot of the last 2000 elements of a chain generated using the autoregressive generating density of Example 11.4.

Example 11.5

This example shows how the Metropolis-Hastings method can be used with an example in Bayesian inference [Roberts, 2000]. This is a genetic linkage example, looking at the genetic linkage of 197 animals. The animals are divided into four categories with frequencies given by

$$Z = (z_1, z_2, z_3, z_4) = (125, 18, 20, 34),$$

with corresponding cell probabilities of

$$\left(\frac{1}{2} + \frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right).$$

From this, we get a posterior distribution of θ, given the data Z, of

$$P(\theta|Z) = \pi(\theta) \propto (2+\theta)^{z_1}(1-\theta)^{z_2+z_3}\theta^{z_4}.$$

We would like to use this to observe the behavior of the parameter θ (i.e., what are likely values for θ) given the data. Note that any constants in the denominator in $\pi(\theta)$ have been eliminated because they cancel in the Metropolis-Hastings sampler. We use the random-walk version where the step is generated by the uniform distribution over the interval $(-a, a)$. Note that we set up a MATLAB **inline** function to get the probability of accepting the candidate point.

```
% Set up the preliminaries.
z1 = 125;
z2 = 18;
z3 = 20;
z4 = 34;
n = 1100;
% Step size for the proposal distribution.
a = 0.1;
% Set up the space to store values.
theta = zeros(1,n);
% Get an inline function to evaluate probability.
strg = '((2+th).^z1).*((1-th).^(z2+z3)).*(th.^z4)';
ptheta = inline(strg,'th','z1','z2','z3','z4');
```

We can now generate the chain as shown below.

```
% Use Metropolis-Hastings random-walk
% where y = theta(i-1) + z
% and z is uniform(-a,a).
% Get initial value for theta.
theta(1) = rand(1);
for i = 2:n
 % Generate from proposal distribution.
 y = theta(i-1) - a + 2*a*rand(1);
 % Generate from uniform.
 u = rand(1);
 alpha = min([ ptheta(y,z1,z2,z3,z4)/...
     ptheta(theta(i-1),z1,z2,z3,z4),1]);
 if u <= alpha
   theta(i) = y;
 else
   theta(i) = theta(i-1);
 end
end
```

We set the burn-in period to 100, so only the last 1000 elements are used to produce the density histogram estimate of the posterior density of θ given in Figure 11.5.

❑

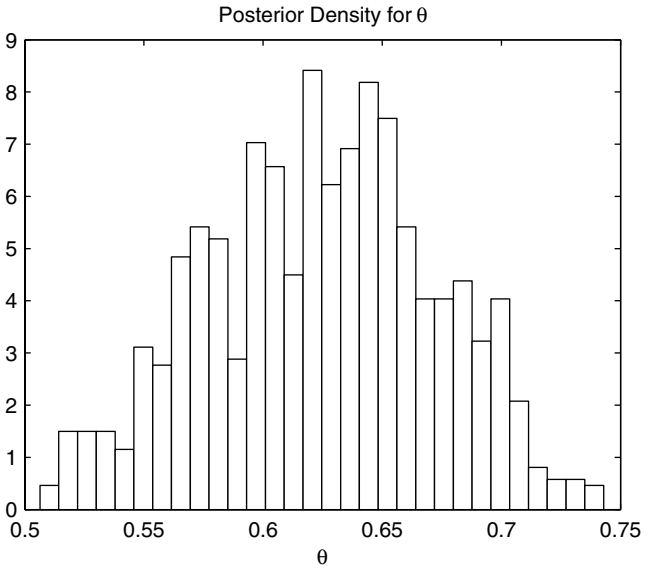

FIGURE 11.5
This shows the density histogram estimate of the posterior density of θ given the observed data.

11.4 The Gibbs Sampler

Although the Gibbs sampler can be shown to be a special case of the Metropolis-Hastings algorithm [Gilks, et al., 1996b; Robert and Casella, 1999], we include it in its own section, because it is different in some fundamental ways. The two main differences between the Gibbs sampler and Metropolis-Hastings are:

1) We always accept a candidate point.
2) We must know the full conditional distributions.

In general, the fact that we must know the full conditional distributions makes the algorithm less applicable.

The Gibbs sampler was originally developed by Geman and Geman [1984], where it was applied to image processing and the analysis of Gibbs distributions on a lattice. It was brought into mainstream statistics through the articles of Gelfand and Smith [1990] and Gelfand, et al. [1990].

In describing the Gibbs sampler, we follow the treatment in Casella and George [1992]. Let's assume that we have a joint density that is given by $f(x, y_1, \ldots, y_d)$, and we would like to understand more about the marginal density. For example, we might want to know the shape, the mean, the variance or some other characteristic of interest.

The marginal density is given by

$$f(x) = \int \ldots \int f(x, y_1, \ldots, y_d) dy_1 \ldots dy_d . \tag{11.10}$$

Equation 11.10 says that to get the marginal distribution, we must integrate over all of the other variables. In many applications, this integration is very difficult (and sometimes impossible) to perform. The Gibbs sampler is a way to get $f(x)$ by simulation. As with the other MCMC methods, we use the Gibbs sampler to generate a sample $X_1, \ldots, X_m$ from $f(x)$ and then use the sample to estimate the desired characteristic of $f(x)$. Casella and George [1992] note that if m is large enough, then any population characteristic can be calculated with the required degree of accuracy.

To illustrate the Gibbs sampler, we start off by looking at the simpler case where the joint distribution is $f(x_1, x_2)$. Using the notation from the previous sections, $\boldsymbol{X}_t$ is a two element vector with elements given by

$$\boldsymbol{X}_t = (X_{t,1}, X_{t,2})$$

We start the chain with an initial starting point of $\boldsymbol{X}_0 = (X_{0,1}, X_{0,2})$. We then generate a sample from $f(x_1, x_2)$ by sampling from the conditional distributions given by $f(x_1 | x_2)$ and $f(x_2 | x_1)$. At each iteration, the elements of the random vector are obtained one at a time by alternately generating values from the conditional distributions. We illustrate this in the procedure given below.

PROCEDURE - GIBBS SAMPLER (BIVARIATE CASE)

1. Generate a starting point $\boldsymbol{X}_0 = (X_{0,1}, X_{0,2})$. Set $t = 0$.
2. Generate a point $X_{t,1}$ from

$$f(X_{t,1} | X_{t,2} = x_{t,2}) .$$

3. Generate a point $X_{t,2}$ from

$$f(X_{t,2} | X_{t+1,1} = x_{t+1,1}) .$$

4. Set $t = t + 1$ and repeat steps 2 through 4.

Note that the conditional distributions are conditioned on the current or most recent values of the other components of $\boldsymbol{X}_t$. Example 11.6 shows how this is done in a simple case taken from Casella and George [1992].

Example 11.6

To illustrate the Gibbs sampler, we consider the following joint distribution

$$f(x, y) \propto \binom{n}{x} y^{x+\alpha-1}(1-y)^{n-x+\beta-1},$$

where $x = 0, 1, \ldots, n$ and $0 \le y \le 1$. Let's say our goal is to estimate some characteristic of the marginal distribution $f(x)$ of X. By ignoring the overall dependence on n, α and β, we find that the conditional distribution $f(x|y)$ is binomial with parameters n and y, and the conditional distribution $f(y|x)$ is a beta distribution with parameters $x+\alpha$ and $n-x+\beta$ [Casella and George, 1992]. The MATLAB commands given below use the Gibbs sampler to generate variates from the joint distribution.

```
% Set up preliminaries.
% Here we use k for the chain length, because n
% is used for the number of trials in a binomial.
k = 1000;     % generate a chain of size 1000
m = 500;      % burn-in will be 500
a = 2;    % chosen
b = 4;
x = zeros(1,k);
y = zeros(1,k);
n = 16;
```

We are now ready to generate the elements in the chain.We start off by generating a starting point.

```
% Pick a starting point.
x(1) = binornd(n,0.5,1,1);
y(1) = betarnd(x(1) + a, n - x(1) + b,1,1);
for i = 2:k
 x(i) = binornd(n,y(i-1),1,1);
 y(i) = betarnd(x(i)+a, n-x(i)+b, 1, 1);
end
```

Note that we do not have to worry about whether or not we will accept the next value in the chain. With Gibbs sampling every candidate is accepted. We can estimate the marginal using the following

$$\hat{f}(x) = \frac{1}{k-m} \sum_{i=m+1}^{k} f(x|y_i).$$

This says that we evaluate the probability conditional on the values of y_i that were generated after the burn-in period. This is implemented in MATLAB as follows:

```
% Get the marginal by evaluating the conditional.
% Use MATLAB's Statistics Toolbox.
% Find the P(X=x|Y's)
fhat = zeros(1,17);
for i = 1:17
 fhat(i) = mean(binopdf(i-1,n,y(500:k)));
end
```

The true marginal probability mass function is [Casella and George, 1992]

$$f(x) = \binom{n}{x} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(\alpha+\beta+n)},$$

for $x = 0, 1, \ldots, n$. We plot the estimated probability mass function along with the true marginal in Figure 11.6. This shows that the estimate is very close to the true function.
❑

Casella and George [1992] and Gelfand and Smith [1990] recommend that K different sequences be generated, each one with length n. Then the last element of each sequence is used to obtain a sample of size K that is approximately independent for large enough K. We do note that there is some disagreement in the literature regarding the utility of running one really long chain to get better convergence to the target distribution or many shorter chains to get independent samples [Gilks, et al., 1996b]. Most researchers in this field observe that one long run would often be used for exploratory analysis and a few moderate size runs is preferred for inferences.

The procedure given below for the general Gibbs sampler is for one chain only. It is easier to understand the basic concepts by looking at one chain, and it is simple to expand the algorithm to multiple chains.

PROCEDURE - GIBBS SAMPLER

1. Generate a starting point $\boldsymbol{X}_0 = (X_{0,1}, \ldots, X_{0,d})$. Set $t = 0$.
2. Generate a point $X_{t,1}$ from

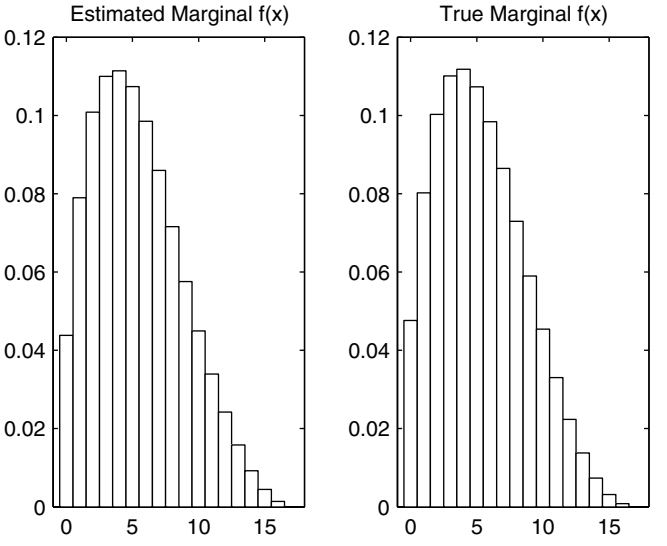

FIGURE 11.6
On the left, we have the estimated probability mass function for the marginal distribution $f(x)$. The mass function on the right is from the true probability mass function. We see that there is close agreement between the two.

$$f(X_{t,1} | X_{t,2} = x_{t,2}, \ldots, X_{t,d} = x_{t,d}) .$$

Generate a point $X_{t,2}$ from

$$f(X_{t,2} | X_{t+1,1} = x_{t+1,1}, X_{t,3} = x_{t,3}, \ldots, X_{t,d} = x_{t,d}) .$$

. . .

Generate a point $X_{t,d}$ from

$$f(X_{t,d} | X_{t+1,1} = x_{t+1,1}, \ldots, X_{t+1,d-1} = x_{t+1,d-1}) .$$

3. Set $t = t + 1$ and repeat steps 2 through 3.

Example 11.7

We show another example of Gibbs sampling as applied to bivariate normal data. Say we have the same model as we had in Example 11.4, where we

wanted to generate samples from a bivariate normal with the following parameters

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}.$$

From Gelman, et al. [1995] we know that $f(x_1|x_2)$ is univariate normal with mean $\mu_1 + \rho(x_2 - \mu_2)$ and standard deviation $1 - \rho^2$. Similarly, $f(x_2|x_1)$ is univariate normal with mean $\mu_2 + \rho(x_1 - \mu_1)$ and standard deviation $1 - \rho^2$. With this information, we can implement the Gibbs sampler to generate the random variables.

```
% Set up constants and arrays.
n = 6000;
xgibbs = zeros(n,2);
rho = 0.9;
y = [1;2];% This is the mean.
sig = sqrt(1-rho^2);
% Initial point.
xgibbs(1,:) = [10 10];
% Start the chain.
for i = 2:n
   mu = y(1) + rho*(xgibbs(i-1,2)-y(2));
   xgibbs(i,1) = mu + sig*randn(1);
   mu = y(2) + rho*(xgibbs(i,1) - y(1));
   xgibbs(i,2) = mu + sig*randn(1);
end
```

Notice that the next element in the chain is generated based on the current values for x_1 and x_2. A scatterplot of the last 2000 variates generated with this method is shown in Figure 11.7.
❑

We return now to our example described at the beginning of the chapter, where we are investigating the hypothesis that there has been a reduction in coal mining disasters over the years 1851 to 1962. To understand this further, we follow the model given in Roberts [2000]. This model assumes that the number of disasters per year follows a Poisson distribution with a mean rate of θ until the *k*-th year. After the *k*-th year, the number of disasters is distributed according to the Poisson distribution with a mean rate of λ. This is represented as

$$\begin{aligned} Y_i &\sim \text{Poisson}(\theta) \qquad i = 1, \ldots, k \\ Y_i &\sim \text{Poisson}(\lambda) \qquad i = k+1, \ldots, n, \end{aligned}$$

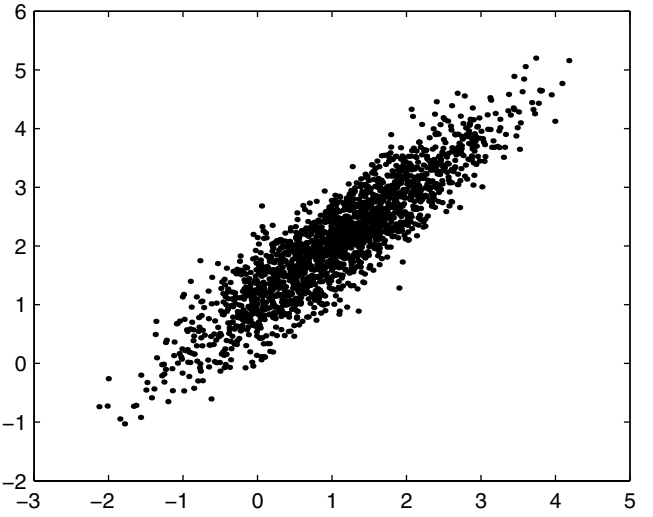

FIGURE 11.7
This is a scatterplot of the bivariate normal variates generated using Gibbs sampling. Note that the results are similar to Figure 11.4.

where the notation '~' means *'is distributed as.'*

A Bayesian model is given by the following

$$\theta \sim \text{Gamma}(a_1, b_1)$$
$$\lambda \sim \text{Gamma}(a_2, b_2)$$
$$b_1 \sim \text{Gamma}(c_1, d_1)$$
$$b_2 \sim \text{Gamma}(c_2, d_2)$$

and the k is discrete uniform over $\{1, \ldots, 112\}$ (since there are 112 years). Note that θ, λ and k are all independent of each other.

This model leads to the following conditional distributions:

$$\theta | Y, \lambda, b_1, b_2, k \sim \text{Gamma}\left(a_1 + \sum_{i=1}^{k} Y_i, k + b_1\right)$$

$$\lambda | Y, \theta, b_1, b_2, k \sim \text{Gamma}\left(a_2 + \sum_{i=k+1}^{n} Y_i, n - k + b_2\right)$$

$$b_1 | Y, \theta, \lambda, b_2, k \sim \text{Gamma}(a_1 + c_1, \theta + d_1)$$
$$b_2 | Y, \theta, \lambda, b_1, k \sim \text{Gamma}(a_2 + c_2, \lambda + d_2)$$
$$f(k|Y, \theta, \lambda, b_1, b_2) = \frac{L(Y;k, \theta, \lambda)}{\sum_{j=1}^{n} L(Y;j, \theta, \lambda)}$$

The likelihood is given by

$$L(Y;k, \theta, \lambda) = \exp\{k(\lambda - \theta)\}(\theta/\lambda)^{\sum_{i=1}^{k} Y_i}.$$

We use Gibbs sampling to simulate the required distributions and examine the results to explore the change-point model. For example, we could look at the posterior densities of θ, λ and k to help us answer the questions posed at the beginning of the chapter.

Example 11.8

A plot of the time series for the **coal** data is shown in Figure 11.8, where we see graphical evidence supporting the hypothesis that a change-point does occur [Raftery and Akman, 1986] and that there has been a reduction in the rate of coal mine disasters over this time period.

We set up the preliminary data needed to implement Gibbs sampling as follows:

```
% Set up preliminaries.
load coal
% y contains number of disasters.
% year contains the year.
n = length(y);
m = 1100;    % number in chain
% The values for the parameters are the same
% as in Roberts[2000].
a1 = 0.5;
a2 = 0.5;
c1 = 0;
c2 = 0;
d1 = 1;
d2 = 1;
theta = zeros(1,m);
lambda = zeros(1,m);
k = zeros(1,n);
```

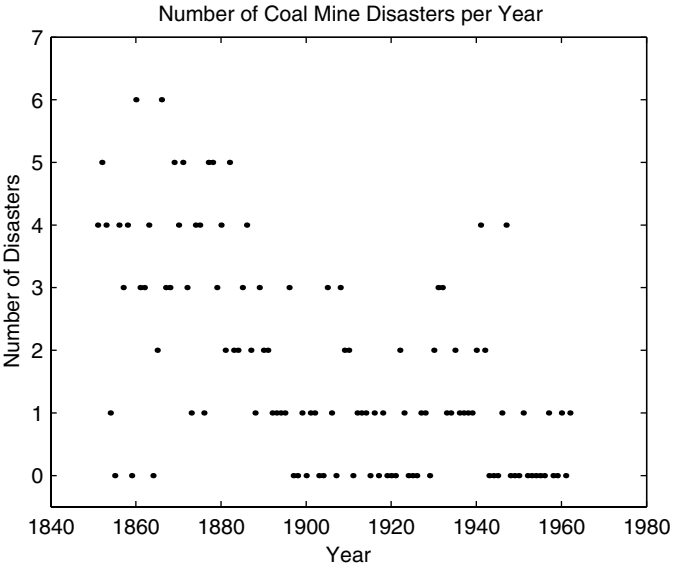

FIGURE 11.8
Time series of the **coal** data. It does appear that there was a reduction in the rate of disasters per year, after a certain year. Estimating that year is the focus of this example.

```
% Holds probabilities for k.
like = zeros(1,n);
```

We are now ready to implement the Gibbs sampling. We will run the chain for 1100 iterations and use a burn-in period of 100.

```
% Get starting points.
k(1) = unidrnd(n,1,1);
% Note that k will indicate an index to the year
% that corresponds to a hypothesized change-point.
theta(1) = 1;
lambda(1) = 1;
b1 = 1;
b2 = 1;
% Start the Gibbs Sampler.
for i = 2:m
   kk = k(i-1);
   % Get parameters for generating theta.
   t = a1 + sum(y(1:kk));
   lam = kk + b1;
   % Generate the variate for theta.
```

```
    theta(i) = gamrnd(t,1/lam,1,1);
    % Get parameters for generating lambda.
    t = a2 + sum(y) - sum(y(1:kk));
    lam = n-kk+b2;
    % Generate the variate for lambda.
    lambda(i) = gamrnd(t,1/lam,1,1);
    % Generate the parameters b1 and b2.
    b1 = gamrnd(a1+c1,1/(theta(i)+d1),1,1);
    b2 = gamrnd(a2+c2,1/(lambda(i)+d2),1,1);
    % Now get the probabilities for k.
    for j = 1:n
       like(j) = exp((lambda(i)-theta(i))*j)*...
          (theta(i)/lambda(i))^sum(y(1:j));
    end
    like = like/sum(like);
    % Now sample the variate for k.
    k(i) = cssample(1:n,like,1);
end
```

The sequences for θ, λ and k are shown in Figure 11.9, where we can see that a burn-in period of 100 is reasonable. In Figure 11.10. we plot the frequencies for the estimated posterior distribution using the generated k variates. We see evidence of a posterior mode at $k = 41$, which corresponds to the year 1891. So, we suspect that the change-point most likely occurred around 1891. We can also look at density histograms for the posterior densities for θ and λ. These are given in Figure 11.11, and they indicate that the mean rate of disasters did decrease after the change-point.

❑

11.5 Convergence Monitoring

The problem of deciding when to stop the chain is an important one and is the topic of current research in MCMC methods. After all, the main purpose of using MCMC is to get a sample from the target distribution and explore its characteristics. If the resulting sequence has not converged to the target distribution, then the estimates and inferences we get from it are suspect.

Most of the methods that have been proposed in the literature are really diagnostic in nature and have the goal of monitoring convergence. Some are appropriate only for Metropolis-Hastings algorithms and some can be applied only to Gibbs samplers.We will discuss in detail one method due to Gelman and Rubin [1992] and Gelman [1996], because it is one of the simplest to understand and to implement. Additionally, it can be used in any of the MCMC algorithms. We also very briefly describe another widely used

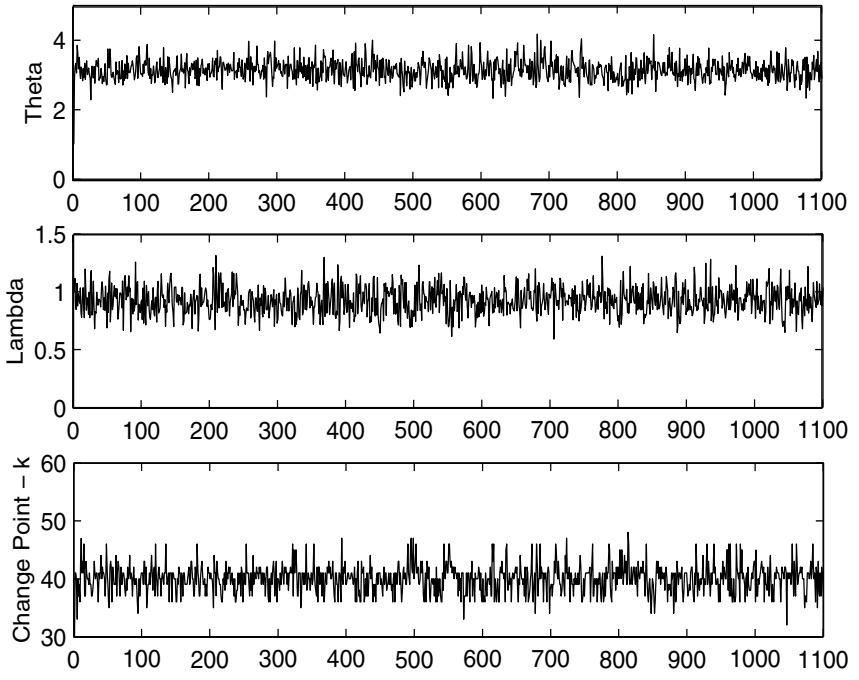

FIGURE 11.9
This shows the sequences that were generated using the Gibbs sampler.

method due to Raftery and Lewis [1992, 1996] that can be employed within the MCMC method. Other papers that review and compare the various convergence diagnostics are Cowles and Carlin [1996], Robert [1995] and Brooks [1998]. Some recent research in this area can be found in Canty [1999] and Brooks and Giudici [2000].

Gelman and Rubin Method

We will use ν to represent the characteristic of the target distribution (mean, moments, quantiles, etc.) in which we are interested. One obvious way to monitor convergence to the target distribution is to run multiple sequences of the chain and plot ν versus the iteration number. If they do not converge to approximately the same value, then there is a problem. Gelman [1996] points out that lack of convergence can be detected by comparing multiple sequences, but cannot be detected by looking at a single sequence.

The Gelman-Rubin convergence diagnostic is based on running multiple chains. Cowles and Carlin [1996] recommend ten or more chains if the target

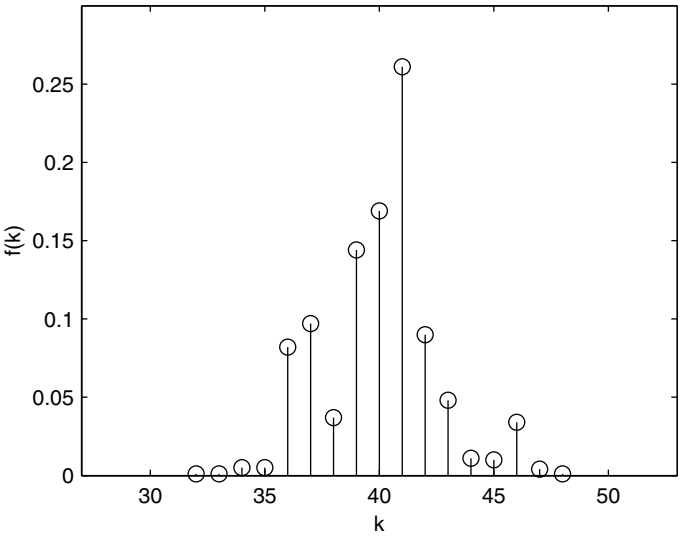

FIGURE 11.10
This is the frequency histogram for the random variables k generated by the Gibbs sampler of Example 11.8. Note the mode at $k = 41$ corresponding to the year 1891.

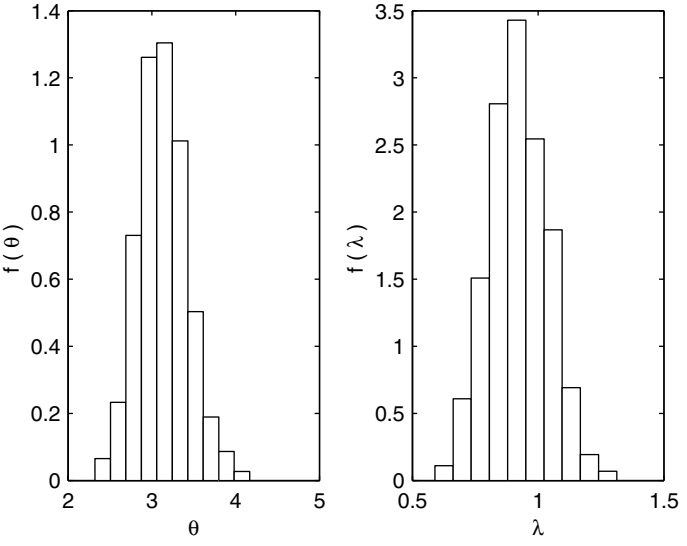

FIGURE 11.11
This figure shows density histograms for the posterior distributions for θ and λ, and there seems to be evidence showing that there was a reduction in the mean rate of disasters per year.

distribution is unimodal. The starting points for these chains are chosen to be widely dispersed in the target distribution. This is important for two reasons. First, it will increase the likelihood that most regions of the target distribution are visited in the simulation. Additionally, any convergence problems are more likely to appear with over-dispersed starting points.

The method is based on the idea that the variance within a single chain will be less than the variance in the combined sequences, if convergence has not taken place. The Gelman-Rubin approach monitors the scalar quantities of interest in the analysis (i.e., ν).

We start off with k parallel sequences of length n starting from over-dispersed points in the target distribution. The between-sequence variance B and the within-sequence W are calculated for each scalar summary ν. We denote the j-th scalar summary in the i-th chain by

$$\nu_{ij}; \qquad i = 1, \ldots, k, \quad j = 1, \ldots, n \ .$$

Thus, the subscript j represents the position in the chain or sequence and i denotes which sequence it was calculated from.

The between-sequence variance is given as

$$B = \frac{n}{k-1} \sum_{i=1}^{k} (\overline{\nu}_{i.} - \overline{\nu}_{..})^2 , \tag{11.11}$$

where

$$\overline{\nu}_{i.} = \frac{1}{n} \sum_{j=1}^{n} \nu_{ij} , \tag{11.12}$$

and

$$\overline{\nu}_{..} = \frac{1}{k} \sum_{i=1}^{k} \overline{\nu}_{i.} . \tag{11.13}$$

Equation 11.12 is the mean of the n values of the scalar summary in the i-th sequence, and Equation 11.13 is the average across sequences.

The within-sequence variance is determined by

$$W = \frac{1}{k} \sum_{i=1}^{k} s_i^2 , \tag{11.14}$$

with

$$s_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(\nu_{ij} - \overline{\nu}_{i.})^2 . \tag{11.15}$$

Note that Equation 11.15 is the sample variance of the scalar summary for the i-th sequence, and Equation 11.14 is the average variance for the k sequences.

Finally, W and B are combined to get an overall estimate of the variance of ν in the target distribution:

$$\hat{\text{var}}(\nu) = \frac{n-1}{n}W + \frac{1}{n}B . \tag{11.16}$$

Equation 11.16 is a conservative estimate of the variance of ν, if the starting points are over-dispersed [Gelman, 1996]. In other words, it tends to over estimate the variance.

Alternatively, the within-sequence variance given by W is an underestimate of the variance of ν. This should make sense considering the fact that finite sequences have not had a chance to travel all of the target distribution resulting in less variability for ν. As n gets large, both $\hat{\text{var}}(\nu)$ and W approach the true variance of ν, one from above and one from below.

The Gelman-Rubin approach diagnoses convergence by calculating

$$\sqrt{\hat{R}} = \sqrt{\frac{\hat{\text{var}}(\nu)}{W}} . \tag{11.17}$$

This is the ratio between the upper bound on the standard deviation of ν and the lower bound. It estimates the factor by which $\hat{\text{var}}(\nu)$ might be reduced by further iterations. The factor given by Equation 11.17 is called the ***estimated potential scale reduction***. If the potential scale reduction is high, then the analyst is advised to run the chains for more iterations. Gelman [1996] recommends that the sequences be run until $\hat{R}$ for all scalar summaries are less than 1.1 or 1.2.

Example 11.9

We return to Example 11.3 to illustrate the Gelman-Rubin method for monitoring convergence. Recall that our target distribution is the univariate standard normal. This time our proposal distribution is univariate normal with $\mu = X_t$ and $\sigma = 5$. Our scalar summary ν is the mean of the elements of the chain. We implement the Gelman-Rubin method using four chains.

```
% Set up preliminaries.
sig = 5;
```

```
% We will generate 500 iterations of the chain.
n = 5000;
numchain = 4;
% Set up the vectors to store the samples.
% This is 4 chains, 5000 samples.
X = zeros(numchain,n);
% This is 4 sequences (rows) of summaries.
nu = zeros(numchain,n);
% Track the rhat for each iteration:
rhat = zeros(1,n);
% Get the starting values for the chain.
% Use over-dispersed starting points.
X(1,1) = -10;
X(2,1) = 10;
X(3,1) = -5;
X(4,1) = 5;
```

The following implements the chains. Note that each column of our matrices **X** and **nu** is one iteration of the chains, and each row contains one of the chains. The **X** matrix keeps the chains, and the matrix **nu** is the sequence of scalar summaries for each chain.

```
% Run the chain.
for j = 2:n
   for i = 1:numchain
      % Generate variate from proposal distribution.
      y = randn(1)*sig + X(i,j-1);
      % Generate variate from uniform.
      u = rand(1);
      % Calculate alpha.
      alpha = normpdf(y,0,1)/normpdf(X(i,j-1),0,1);
      if u <= alpha
         % Then set the chain to the y.
         X(i,j) = y;
      else
         X(i,j) = X(i,j-1);
      end
   end
   % Get the scalar summary - means of each row.
   nu(:,j) = mean(X(:,1:j)')';
   rhat(j) = csgelrub(nu(:,1:j));
end
```

The function **csgelrub** will return the estimated $\hat{R}$ for a given set of sequences of scalar summaries. We plot the four sequences for the summary statistics of the chains in Figure 11.12. From these plots, we see that it might be reasonable to assume that the sequences have converged, since they are

getting close to the same value in each plot. In Figure 11.13, we show a plot of $\hat{R}$ for each iteration of the sequence. This seems to confirm that the chains are getting close to convergence. Our final value of $\hat{R}$ at the last iteration of the chain is 1.05.
❑

One of the advantages of the Gelman-Rubin method is that the sequential output of the chains does not have to be examined by the analyst. This can be difficult, especially when there are a lot of summary quantities that must be monitored. The Gelman-Rubin method is based on means and variances, so it is especially useful for statistics that approximately follow the normal distribution. Gelman, et al. [1995] recommend that in other cases, extreme quantiles of the between and within sequences should be monitored.

Raftery and Lewis Method

We briefly describe this method for two reasons. First, it is widely used in applications. Secondly, it is available in MATLAB code through the Econometrics Toolbox (see Section 11.6 for more information) and in Fortran from StatLib. So, the researcher who needs another method besides the one of Gelman and Rubin is encouraged to download these and try them. The article by Raftery and Lewis [1996] is another excellent resource for information on the theoretical basis for the method and for advice on how to use it in practice.

This technique is used to detect convergence of the chain to the target distribution and also provides a way to bound the variance of the estimates obtained from the samples. To use this method, the analyst first runs one chain of the Gibbs sampler for $N_{\min}$. This is the minimum number of iterations needed for the required precision, given that the samples are independent. Using this chain and other quantities as inputs (the quantile to be estimated, the desired accuracy, the probability of getting that accuracy, and a convergence tolerance), the Raftery-Lewis method yields several useful values. Among them are the total number of iterations needed to get the desired level of accuracy and the number of points in the chain that should be discarded (i.e., the burn-in).

11.6 MATLAB Code

The Statistics Toolbox for MATLAB does not provide functions that implement MCMC methods, but the pieces (i.e., evaluating probability density functions and generating random variables) are there for the analyst to easily code up the required simulations. Also, the examples given in this text can be adapted to fit most applications by simply changing the proposal and target

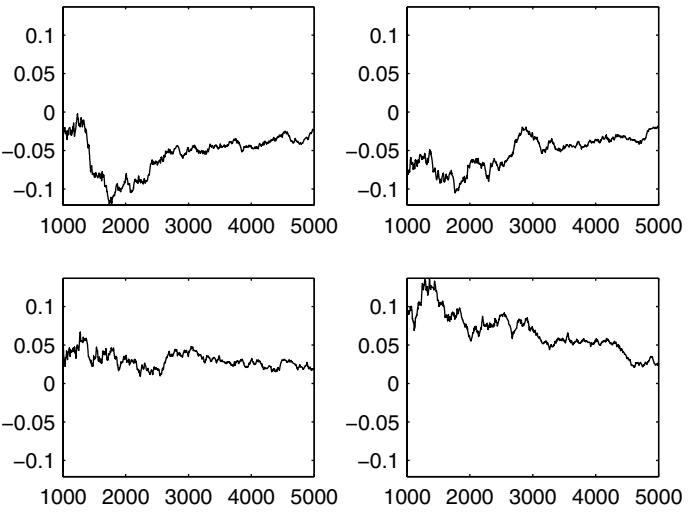

FIGURE 11.12
Here are the sequences of summary statistics in Example 11.9. We are tracking the mean of sequences of variables generated by the Metropolis-Hastings sampler. The target distribution is a univariate standard normal. It appears that the sequences are close to converging, since they are all approaching the same value.

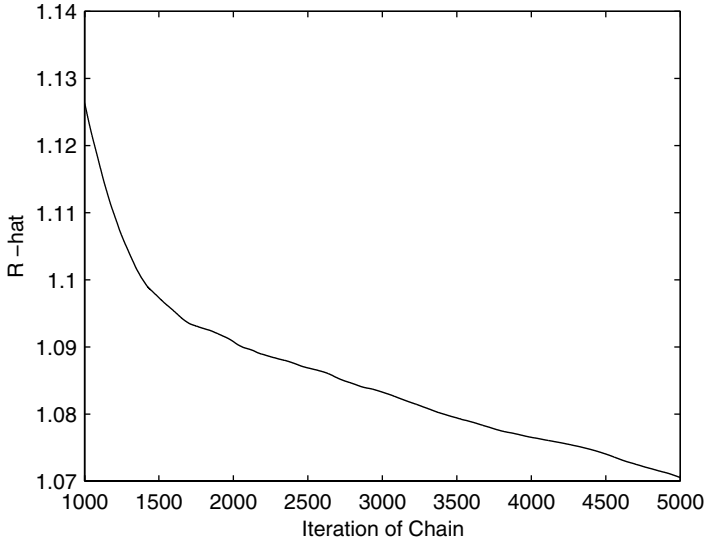

FIGURE 11.13
This sequence of values for $\hat{R}$ indicates that it is very close to one, showing near convergence.

distributions. There is an Econometrics Toolbox that contains M-files for the Gibbs sampler and the Raftery-Lewis convergence diagnostic. The software can be freely downloaded at **www.spatial-econometrics.com**. Extensive documentation for the procedures in the Econometrics Toolbox is also available at the website. The Raftery-Lewis method for S-plus and Fortran can be downloaded at:

- S-plus: **http://lib.stat.cmu.edu/S/gibbsit**
- Fortran: **http://lib.stat.cmu.edu/general/gibbsit**

There are several user-contributed M-files for MCMC available for download at The MathWorks website:

ftp.mathworks.com/pub/contrib/v5/stats/mcmc/

For those who do not use MATLAB, another resource for software that will do Gibbs sampling and Bayesian analysis is the BUGS (Bayesian Inference Using Gibbs Sampling) software. The software and manuals can be downloaded at **www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml**.

In the Computational Statistics Toolbox, we provide an M-file function called **csgelrub** that implements the Gelman-Rubin diagnostic. It returns $\hat{R}$ for given sequences of scalar summaries. We also include a function that implements a demo of the Metropolis-Hastings sampler where the target distribution is standard bivariate normal. This runs four chains, and the points are plotted as they are generated so the user can see what happens as the chain grows. The M-file functions pertaining to MCMC that we provide are summarized in Table 11.1.

TABLE 11.1

List of Functions from Chapter 11 Included in the Computational Statistics Toolbox

Purpose	MATLAB Function
Gelman-Rubin convergence diagnostic given sequences of scalar summaries	**csgelrub**
Graphical demonstration of what happens in the Metropolis-Hastings sampler	**csmcmcdemo**

11.7 Further Reading

For an excellent introduction to Markov chain Monte Carlo methods, we recommend the book *Markov Chain Monte Carlo in Practice* [Gilks, et al., 1996b]. This contains a series of articles written by leading researchers in the area and describes most aspects of MCMC from the theoretical to the practical. For a complete theoretical treatment of MCMC methods and many examples, the reader is referred to Robert and Casella [1999]. This book also contains a description of many of the hybrid MCMC methods that have been developed. The text by Tanner [1996] provides an introduction to computational algorithms for Bayesian and likelihood inference.

Most recent books on random number generation discuss the Metropolis-Hastings sampler and the Gibbs sampler. Gentle [1998] has a good discussion of MCMC methods and includes some examples in MATLAB. Ross [1997] has a chapter on MCMC and also discusses the connection between Metropolis-Hastings and simulated annealing. Ross [2000] also covers the topic of MCMC.

The monograph by Lindley [1995] gives an introduction and review of Bayesian statistics. For an overview of general Markov chain theory, see Tierney [1996], Meyn and Tweedie [1993] or Norris [1997]. If the reader would like more information on Bayesian data analysis, then the book *Bayesian Data Analysis* [Gelman, et al., 1995] is a good place to start. This text also contains some information and examples about the MCMC methods discussed in this chapter. Most of these books also include information on Monte Carlo integration methods, including importance sampling and variance reduction.

Besides simulated annealing, a connection between MCMC methods and the finite mixtures EM algorithm has been discussed in the literature. For more information on this, see Robert and Casella [1999]. There is also another method that, while not strictly an MCMC method, seems to be grouped with them. This is called Sampling Importance Resampling [Rubin, 1987, 1988]. A good introduction to this can be found in Ross [1997], Gentle [1998] and Albert [1993].

Exercises

11.1. The von Mises distribution is given by

$$f(x) = \frac{1}{2\pi I_0(b)} e^{b\cos(x)} \qquad -\pi \le x \le \pi,$$

where I_0 is the modified Bessel function of the first kind and order zero. Letting $b = 3$ and a starting point of 1, use the Metropolis random-walk algorithm to generate 1000 random iterations of the chain. Use the uniform distribution over the interval $(-1, 1)$ to generate steps in the walk. Plot the output from the chain versus the iteration number. Does it look like you need to discard the initial values in the chain for this example? Plot a histogram of the sample [Gentle, 1998].

11.2. Use the Metropolis-Hastings algorithm to generate samples from the beta distribution. Try using the uniform distribution as a candidate distribution. Note that you can simplify by canceling constants.

11.3. Use the Metropolis-Hastings algorithm to generate samples from the gamma distribution. What is a possible candidate distribution? Simplify the ratio by canceling constants.

11.4. Repeat Example 11.3 to generate a sample of standard normal random variables using different starting values and burn-in periods.

11.5. Let's say that $X_{,1}$ and $X_{,2}$ have conditional distributions that are exponential over the interval $(0, B)$, where B is a known positive constant. Thus,

$$f(x_{,1}|x_{,2}) \propto x_{,2} e^{-x_{,2}x_{,1}} \qquad 0 < x_{,1} < B < \infty$$

$$f(x_{,2}|x_{,1}) \propto x_{,1} e^{-x_{,1}x_{,2}} \qquad 0 < x_{,2} < B < \infty$$

Use Gibbs sampling to generate samples from the marginal distribution $f(x_{,1})$. Choose your own starting values and burn-in period. Estimate the marginal distribution. What is the estimated mean, variance, and skewness coefficient for $f(x_{,1})$? Plot a histogram of the samples obtained after the burn-in period and the sequential output. Start multiple chains from over-dispersed starting points and use the Gelman-Rubin convergence diagnostics for the mean, variance and skewness coefficient [Casella and George, 1992].

11.6. Explore the use of the Metroplis-Hastings algorithm in higher dimensions. Generate 1000 samples for a trivariate normal distribution cen-

tered at the origin and covariance equal to the identity matrix. Thus, each coordinate direction should be a univariate standard normal distribution. Use a trivariate normal distribution with covariance matrix $\mathbf{\Sigma} = 9 \cdot \mathbf{I}$, (i.e., 9's are along the diagonal and 0's everywhere else) and mean given by the current value of the chain $\mathbf{x}_t$. Use $x_{0,i} = 10$, $i = 1, \ldots, 3$ as the starting point of the chain. Plot the sequential output for each coordinate. Construct a histogram for the first coordinate direction. Does it look like a standard normal? What value did you use for the burn-in period? [Gentle, 1998.]

11.7. A joint density is given by

$$f(x_{,1}, x_{,2}, x_{,3}) = C\exp\{-(x_{,1} + x_{,2} + x_{,3} + x_{,1}x_{,2} + x_{,1}x_{,3} + x_{,2}x_{,3})\},$$

where $x_{,i} > 0$. Use one of the techniques from this chapter to simulate samples from this distribution and use them to estimate $E[X_{,1}X_{,2}X_{,3}]$. Start multiple chains and track the estimate to monitor the convergence [Ross, 1997].

11.8. Use Gibbs sampling to generate samples that have the following density

$$f(x_{,1}, x_{,2}, x_{,3}) = kx_{,1}^4 x_{,2}^3 x_{,3}^2 (1 - x_{,1} - x_{,2} - x_{,3})$$

where $x_{,i} > 0$ and $x_{,1} + x_{,2} + x_{,3} < 1$. Let $B(a, b)$ represent a beta distribution with parameters a and b. We can write the conditional distributions as

$$\begin{aligned} X_{,1}|X_{,2}, X_{,3} &\sim (1 - X_{,2} - X_{,3})Q \qquad & Q &\sim B(5, 2) \\ X_{,2}|X_{,1}, X_{,3} &\sim (1 - X_{,1} - X_{,3})R \qquad & R &\sim B(4, 2) \\ X_{,3}|X_{,1}, X_{,2} &\sim (1 - X_{,1} - X_{,2})S \qquad & S &\sim B(3, 2) \end{aligned}$$

where the notation $Q \sim B(a, b)$ means Q is from a beta distribution. Plot the sequential output for each $x_{,i}$ [Arnold, 1993].

11.9. Let's say that we have random samples $Z_1, \ldots, Z_n$ that are independent and identically distributed from the normal distribution with mean θ and variance 1. In the notation of Equation 11.1, these constitute the set of observations D. We also have a prior distribution on θ such that

$$P(\theta) \propto \frac{1}{1 + \theta^2},$$

We can write the posterior as follows

$$P(\theta|D) \propto P(\theta)L(\theta;D) = \frac{1}{1+\theta^2} \times \exp\left\{\frac{-n(\theta-\bar{z})^2}{2}\right\}.$$

Let the true mean be $\theta = 0.06$ and generate a random sample of size $n = 20$ from the normal distribution to obtain the z_i. Use Metropolis-Hastings to generate random samples from the posterior distribution and use them to estimate the mean and the variance of the posterior distribution. Start multiple chains and use the Gelman-Rubin diagnostic method to determine when to stop the chains.

11.10. Generate a set of $n = 2000$ random variables for the bivariate distribution given in Example 11.4 using the technique from Chapter 4. Create a scatterplot of these data and compare to the set generated in Example 11.4.

11.11. For the bivariate distribution of Example 11.4, use a random-walk generating density $(\boldsymbol{Y} = \boldsymbol{X}_t + \boldsymbol{Z})$ where the increment random variable $\boldsymbol{Z}$ is distributed as bivariate uniform. Generate a sequence of 6000 elements and construct a scatterplot of the last 2000 values. Compare to the results of Example 11.4.

11.12. For the bivariate distribution of Example 11.4, use a random-walk generating density $(\boldsymbol{Y} = \boldsymbol{X}_t + \boldsymbol{Z})$ where the increment random variables $\boldsymbol{Z}$ are bivariate normal with mean zero and covariance

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.4 \end{bmatrix}.$$

Generate a sequence of 6000 elements and construct a scatterplot of the last 2000 values. Compare to the results of Example 11.4.

11.13. Use the Metropolis-Hastings sampler to generate random samples from the lognormal distribution

$$f(x) = \frac{1}{x\sqrt{2\pi}} \exp\left\{-\frac{(\ln x)^2}{2}\right\}$$

$$f(x) \propto \frac{1}{x} \exp\left\{-\frac{(\ln x)^2}{2}\right\}.$$

Use the independence sampler and the gamma as a proposal distribution, being careful about the tails. Plot the sample using the density histogram and superimpose the true probability density function to ensure that your random variables are from the desired distribution.